

REMARKS**Status of the Claims**

Claims 1-26 are currently present in the Application, and claims 1, 10, 15, 24, 25, and 26 are independent claims. Claims 1, 3, 7-10, 12, 14-17, and 21-26 have been amended to correct potential antecedent basis issues with these claims. In addition, independent claims 15 and 26 have been amended to prevent any future issues with regard to 35 U.S.C. § 101. In particular, the preambles of independent claims 15 and 26 have been amended to clarify that the computer readable media includes instructions for execution by a computer which, when executed by the computer, cause the computer to perform the claimed method. Support for the amendments to the computer program product claims is found, for example, in Applicant's specification on page 16, lines 1-18. Claims 16, 17, 21, 22, and 23 have been amended to correspond to the amendments in independent claim 15. No claims have been canceled or added in this Response, and no new matter has been added as a result of the amendments.

Claim Rejections – Alleged Obviousness Under 35 U.S.C. § 103

Claims 1-26 stand rejected under 35 U.S.C. § 103(a) as being obvious over Sayag, U.S. Patent No. 6,898,602 (hereinafter Sayag), in view of Kundu et al., U.S. Publication No. 2004/0210577 (hereinafter Kundu), and further in view of Kolawa et al., U.S. Patent No. 5,842,019 (hereinafter Kolawa). Applicant respectfully traverses the rejections under 35 U.S.C. § 103.

A. There Is No Motivation To Combine Sayag, Kundu, And Kolawa

MPEP § 706.02(j) states, inter alia:

To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings.

MPEP § 2143.01 states, inter alia (emphasis added):

FACT THAT REFERENCES CAN BE COMBINED OR MODIFIED IS NOT SUFFICIENT TO ESTABLISH *PRIMA FACIE* OBVIOUSNESS

The mere fact that references can be combined or modified does not render the resultant combination obvious unless the prior art also suggests the desirability of the combination. *In re Mills*, 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990) (Claims were directed to an apparatus for producing an aerated cementitious composition by drawing air into the cementitious composition by driving the output pump at a capacity greater than the feed rate. The prior art reference taught that the feed means can be run at a variable speed, however the court found that this does not require that the output pump be run at the claimed speed so that air is drawn into the mixing chamber and is entrained in the ingredients during operation. Although a prior art device "may be capable of being modified to run the way the apparatus is claimed, there must be a suggestion or motivation in the reference to do so." 916 F.2d at 682, 16 USPQ2d at 1432.). See also *In re Fritch*, 972 F.2d 1260, 23 USPQ2d 1780 (Fed. Cir. 1992) (flexible landscape edging device which is conformable to a ground surface of varying slope not suggested by combination of prior art references).

Regarding the three references cited in the Office Action, Sayag discloses the "use of garbage collection in order to determine the exact amount of memory that is consumed by a running application at any point of its execution" (see Sayag, Abstract). Kolawa discloses "dynamically detecting leaked memory space in a computer program" (see Kolawa, Abstract). Both Sayag and Kolawa deal with the general technology area of runtime memory space. Sayag is concerned with determining the amount of memory space used by a runtime application, while Kolawa is concerned with detecting memory space leaks during runtime. Kundu, on the other hand, is concerned with a completely different technology area. Kundu discloses "[t]echniques for making light-weight checkpoints in logs of streams of transactions" (see Kundu, Abstract). Kundu uses the redo logs that are typically used in database systems to log transactions, and purports to increase the usefulness of these redo logs for purposes such as data mining and

replication of transactions. As discussed in detail in Kundu, a logical redo log is made from a physical redo log. Light-weight checkpoints are made and used in the logical redo log, and these light-weight checkpoints are used in data mining and replication (Kundu, paragraph [0041]). Kundu is not concerned with, and indeed does not even address, the issue of memory space as used by runtime applications. Kundu is in a completely different technology area from Sayag and Kolawa. Kundu is concerned with database and DBMS technology, whereas Sayag and Kolawa deal with runtime memory space. Other than the fact that all three references have to do with the very general area of computer science, there is simply no justification to combine Kundu with either Sayag or Kolawa.

It appears that the Office Action improperly uses Applicant's claims as "guideposts" in selecting the references and simply concludes that it is "obvious" to combine the references. In doing so, Applicant asserts that the Office Action uses impermissible hindsight in combining Kundu with Sayag and Kolawa in order to support a rejection of Applicant's claims. As an example, in the rejection of claim 1, the Office Action cites Sayag as disclosing Applicant's first element ("reading one or more variables . . ."). The Office Action then cites Kundu as disclosing Applicant's second element ("identifying a program statement . . .") and the first part of Applicant's third element ("inserting a nullification statement . . ."). The Office Action then cites Kolawa as disclosing the remainder of Applicant's third element ("the nullification statement adapted to nullify . . ."). As discussed in further detail below, the Office Action cites Kundu as disclosing the step of "inserting a nullification statement" and then cites Kolawa as disclosing the definition of the nullification statement, i.e. "the nullification statement adapted to nullify the selected variable." Dissecting Applicant's claims in this manner and then using hindsight to pick and choose a variety of references to read on selected sections of Applicant's claim elements is not permissible. As stated in MPEP § 2143.03, "[t]he mere fact that references can be combined or modified does not render the resultant combination obvious unless the prior art also suggests the desirability of the combination" (emphasis added).

In this case, the prior art simply does not suggest the desirability of combining these references. Applicant submits that the Office Action fails to satisfy the burden set forth in MPEP §§ 706.02(j) and 2143.03 in support of an obviousness objection, particularly because there is no motivation to combine the references. Thus, Applicant contends that the Office Action uses impermissible hindsight in rejecting Applicant's claims. For the reasons set forth above, Applicant respectfully submits that claims 1-26 are not obvious, and are therefore patentable over Sayag in view of Kundu and Kolawa.

B. The Cited References Do Not Teach Or Suggest All The Limitations Of Applicant's Claims

To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). "All words in a claim must be considered in judging the patentability of that claim against the prior art." *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970). If an independent claim is nonobvious under 35 U.S.C. 103, then any claim depending therefrom is nonobvious. *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988). Manual of Patent Examining Procedure § 2143.03. Applicant respectfully submits that none of the cited references, either alone or in combination, teaches or suggests all the elements of Applicant's claims.

Using independent claim 1 as an exemplary claim, Applicant teaches and claims the following:

- reading one or more variables included in one or more activation records included in the middleware computer program;
- identifying a program statement in the middleware computer program where a selected variable is last used; and

- inserting a nullification statement after the identified program statement, the nullification statement adapted to nullify the selected variable.

The Office Action asserts that Sayag discloses the limitation of reading variables included in activation records, but correctly notes that Sayag does not teach or suggest Applicant's "identifying" and "inserting" limitations (see Office Action, page 3). The Office Action then cites Kundu as disclosing identifying a program statement in a computer program where a variable is last used, and inserting a nullification statement after the identified program statement (see Office Action, page 4). Applicant respectfully disagrees. As discussed above, Kundu discloses "[t]echniques for making light-weight checkpoints in logs of streams of transactions" (see Kundu, Abstract). Kundu uses the redo logs that are typically used in database systems to log transactions, and purports to increase the usefulness of these redo logs for purposes such as data mining and replication of transactions. As discussed in detail in Kundu, a logical redo log is made from a physical redo log. Light-weight checkpoints are made and used in the logical redo log, and these light-weight checkpoints are used in data mining and replication (Kundu, paragraph [0041]).

The Office Action cites Kundu at paragraph [0051] as teaching the "identifying" step of Applicant's independent claims. The cited section of Kundu discusses "the information needed to make a logical redo log from a physical redo log" (Kundu, paragraph [0050]). This information includes various fields, such as session#, client#, server#, session_name, etc. The Office Action asserts that Kundu's "checkpoint_scn" is somehow equivalent to "identifying a program statement in the middleware computer program where a selected variable is last used." Applicant respectfully disagrees. Kundu's checkpoint_scn is the system change number, SCN, of the most recently-made checkpoint in the physical redo log. As known to those skilled in the art, a checkpoint is a saved state of a program and its data, which can be used to restart the program at the point at which the checkpoint occurred. As explained in Kundu, "locations in physical

redo logs are identified by system change numbers" (Kundu, paragraph [0051]). These system change numbers are used to identify locations within a physical redo log. Applicant fails to see how a system change number can be said to read on Applicant's step of "identifying a program statement in the middleware computer program where a selected variable is last used." Kundu is not at all concerned with determining where, within a program, a variable is last used. The physical redo logs in Kundu are concerned with taking a snapshot of a database program at a particular point in time. The variables used within that time frame may or may not continue to be used after the checkpoint is taken. Kundu simply is not concerned with determining where a variable is last used.

The cited section of Kundu at paragraph [0066] discusses producing a logical redo log. Checkpoints are made in the logical redo log by inserting checkpoint logical change records, LCRs, at the proper locations in the logical redo log. The Office Action asserts that inserting a logical change record into a logical redo log is equivalent to "inserting a nullification statement after the identified program statement," but then admits that Kundu does not disclose "the nullification statement adapted to nullify the selected variable." Applicant respectfully reminds the Examiner that MPEP § 2143.03 states that all the limitations of a claim must be considered. As stated in MPEP § 2143.03:

2143.03 All Claim Limitations Must Be Taught or Suggested

To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). "All words in a claim must be considered in judging the patentability of that claim against the prior art." *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970). If an independent claim is nonobvious under 35 U.S.C. 103, then any claim depending therefrom is nonobvious. *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988).

Applicant clearly claims that the nullification statement that is inserted after the identified program statement is meant "to nullify the selected variable." Kundu does not

teach or suggest a nullification statement that nullifies a variable. Kundu discusses inserting logical change records into a logical redo log. A logical change record is not the same as a nullification statement. Even assuming, for the sake of argument (and Applicant does not agree that this is the case) that a logical change record could somehow be said to be equivalent to a nullification statement, the logical change records disclosed by Kundu do not nullify a variable. Kundu does not teach or suggest nullifying a variable, and Kundu certainly does not teach or suggest nullifying a variable by inserting a nullification statement after a program statement in which the variable is last used. Further, the Office Action admits that Kundu's logical change records do not nullify variables (see Office Action, page 5). Applicant specifically teaches and claims a nullification statement that nullifies a selected variable, and yet the Office Action admits that Kundu does not teach this aspect of Applicant's claim.

The Office Action then uses yet another reference, i.e. Kolawa, to disclose "the nullification statement adapted to nullify the selected variable." However, the cited section of Kolawa at col. 5, lines 40-61, discusses a routine for performing leak searching. Allocated memory space is tracked using reference counting. A reference count is assigned to a pointer, and if the reference count becomes zero, it indicates that a memory leak may exist. Setting a reference count to zero does not nullify any variables, nor does it nullify the pointer. As clearly shown in Figure 7 of Kolawa, when the reference count equals zero, it indicates that there may be a memory leak, and thus it is time to do a memory search in order to determine if there is a memory leak. Using a reference count to determine that a potential memory leak exists is simply not the same as "inserting a nullification statement after the identified program statement, the nullification statement adapted to nullify the selected variable," as taught and claimed by Applicant. Kolawa simply has nothing to do with nullifying variables.

Because none of the cited references, either alone or in combination, teach or suggest all the elements of independent claim 1, Applicant respectfully submits that independent claim 1 is patentable over the cited references. Independent claims 10, 15, and 24-26 include limitations similar to those in independent claim 1, and are

therefore patentable for at least the reasons discussed above with regard to claim 1. Therefore, Applicants respectfully submits that independent claims 1, 10, 15, and 24-26, and the claims which depend from them, are patentable over Sayag in view of Kundu and Kolawa.

Notwithstanding the patentability of independent claims 24-26 based on the above discussion, Applicant would like to discuss these claims in further detail. Using independent claim 24 as an exemplary claim, Applicant claims the following:

- reading one or more variables included in one or more activation records included in the middleware computer program;
- identifying a program statement in the middleware computer program where a selected variable is last used;
- inserting a nullification statement after the identified program statement, the nullification statement adapted to nullify the selected variable;
- writing a plurality of program statements, including the identified program statement, to a resulting code file; and
- writing the nullification statement to the resulting code file in a position subsequent to the identified program statement.

As discussed above, the prior art does not teach or suggest the “identifying” and “inserting” steps of Applicant’s independent claims. Applicant further submits that none of the prior art, either alone or in combination, teaches or suggests “writing the nullification statement to the resulting code file in a position subsequent to the identified program statement,” as taught and claimed by Applicants. The Office Action cites various sections of Sayag as disclosing this element. However, the Office Action also

admits that Sayag does not disclose “identifying a program statement in the middleware computer program where a selected variable is last used,” and “inserting a nullification statement after the identified program statement, the nullification statement adapted to nullify the selected variable” (see Office Action, page 3). Applicant is at a loss to understand how, if Sayag does not teach identifying a program statement where a selected variable is last used or inserting a nullification statement after the identified program statement, Sayag can then be said to teach writing this nullification statement to a resulting code file in a position subsequent to the identified program statement. The cited sections of Sayag may discuss writing program statements to a file, but they do not discuss writing a nullification statement in a position subsequent to an identified program statement, where the identified program statement is specifically claimed to be the program statement in which a variable is last used, as taught and claimed by Applicant.

Independent claims 25 and 26 include limitations similar to those found in independent claim 24. For the reasons set forth above, Applicant respectfully submits that independent claims 24-26 are patentable over Sayag in view of Kundu and Kolawa.

Conclusion

As a result of the foregoing, it is asserted by Applicant that the remaining claims in the Application are in condition for allowance, and Applicant respectfully requests an early allowance of such claims.

Applicant respectfully request that the Examiner contact the Applicant's attorney listed below if the Examiner believes that such a discussion would be helpful in resolving any remaining questions or issues related to this Application.

Respectfully submitted,

By /Leslie A. Van Leeuwen, Reg. No. 42,196/

Leslie A. Van Leeuwen, Reg. No. 42,196

Van Leeuwen & Van Leeuwen

Attorneys for Applicant

Telephone: (512) 301-6738

Facsimile: (512) 301-6742